Examiner-Initiated Interview Summary	**Application No.** 09/964,237	**Applicant(s)** BATES ET AL.
	Examiner J. Derek Rutten	**Art Unit** 2192

All Participants: **Status of Application:** *Allowed*

(1) *J. Derek Rutten*. (3) _____.

(2) *Gero G. McClellan, Reg. No. 44227*. (4) _____.

Date of Interview: *15 December 2005* **Time:** _____

Type of Interview:
- ☒ Telephonic
- ☐ Video Conference
- ☐ Personal (Copy given to: ☐ Applicant ☐ Applicant's representative)

Exhibit Shown or Demonstrated: ☐ Yes ☒ No
If Yes, provide a brief description: .

Part I.

Rejection(s) discussed:
Claims 1 and 5 under 103(a)

Claims discussed:
1, 2, 4, 5-8, and 14-20

Prior art documents discussed:
None

Part II.

SUBSTANCE OF INTERVIEW DESCRIBING THE GENERAL NATURE OF WHAT WAS DISCUSSED:
See Continuation Sheet

Part III.

☒ It is not necessary for applicant to provide a separate record of the substance of the interview, since the interview directly resulted in the allowance of the application. The examiner will provide a written summary of the substance of the interview in the Notice of Allowability.

☐ It is not necessary for applicant to provide a separate record of the substance of the interview, since the interview did not result in resolution of all issues. A brief summary by the examiner appears in Part II above.

[illegible]
(Examiner/SPE Signature)

(Applicant/Applicant's Representative Signature – if appropriate)

Continuation of Substance of Interview including description of the general nature of what was discussed: The Examiner indicated that claims 4 and 8 appeared to contain allowable subject matter but that claims 5-8 and 14-20 appeared to contain 101 issues, and claims 2 and 6 contained a 112 2nd issue. The Examiner suggested that an Examiner's Amendment to incorporate the subject matter of claims 4 and 8 into the independent claims 1 and 5, respectively, and further amending at least claims 5 and 14 to overcome the 101 issues would put the application in condition for allowance. Mr. McClellan agreed to draw up the discussed amendments and send a copy to the Examiner as a template for the amendment.

PATENT
App. Ser. No.: 09/964,237
Atty. Dkt. No. ROC920010221US1
PS Ref. No.: IBMK10221

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of: Bates et al.	§	
	§	Group Art Unit: 2192
Serial No.: 09/964,237	§	
	§	Derek Rutten
Confirmation No.: 1864	§	Examiner: ~~Chrystine Pham~~
	§	
Filed: September 26, 2001	§	
	§	
For: DYNAMIC SETTING OF BREAKPOINT COUNT ATTRIBUTES	§	
	§	

MAIL STOP AF
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

UNOFFICIAL SUPPLEMENTAL RESPONSE TO FINAL OFFICE ACTION DATED OCTOBER 19, 2005

The Examiner is kindly asked to enter these amendments by way of Examiner's amendment.

LISTING OF THE CLAIMS:

1. (Currently Amended) A method of debugging an application in a debugging environment comprising the application and a debugger program, the method comprising:

automatically counting a number of times each breakpoint in the application is encountered while the application is executing during a test run, wherein the number is increasing and, at any time during the test run, always reflects a current number of times a given breakpoint has been encountered during the test run and wherein counting the number is not limited by a predetermined number representative of a desired number of encounters of the given breakpoint; and storing the number for each breakpoint in a memory space for use in a subsequent run; <u>wherein automatically counting comprises, for each breakpoint: (i) incrementing a breakpoint-specific counter each time a breakpoint associated with the breakpoint-specific counter is encountered in a particular code segment; and (ii) resetting the breakpoint-specific counter each time the particular code segment is entered.</u>

2. (Currently Amended) The method of claim 1, further comprising, while the application is stopped, receiving a user-input request to uninterruptedly execute the application until a user-specified breakpoint is encountered some number of times, N-X, where N is a stored number of times the user-specified breakpoint was encountered during the test run and X is a value equal to or greater than zero<u> and less than N</u>.

3. (Original) The method of claim 1, further comprising, after the application is stopped at a location in response to a last breakpoint encounter of a particular breakpoint encountered Y number of times at the last breakpoint encounter, receiving a user-input request to uninterruptedly execute the application until the application is again stopped at the location in response to encountering the particular breakpoint Y number of times.

4. (Canceled) The method of claim 1, wherein automatically counting comprises, for each breakpoint:

incrementing a breakpoint-specific counter each time a breakpoint associated with the breakpoint-specific counter is encountered in a particular code segment; and resetting the breakpoint-specific counter each time the code segment is entered.

5. (Currently Amended) A tangible computer readable storage medium containing a debug program which, when executed by a computer configured with an application being debugged during a debug session, performs breakpoint counter operations, the debug program comprising:

counting instructions for automatically counting a number of times each breakpoint in the application is encountered, wherein the number is increasing and, at any time, always reflects a current number of times a given breakpoint has been encountered and wherein counting the number is not limited by a predetermined number representative of a desired number of encounters of the given breakpoint; and storing instructions for storing the number for each breakpoint in a memory space for use in a subsequent run; wherein the counting instructions, when executed: (i) increment a breakpoint-specific counter each time a breakpoint associated with the breakpoint-specific counter is encountered in a particular code segment; and (ii) reset the breakpoint-specific counter each time the particular code segment is entered.

6. (Currently Amended) The tangible computer readable storage medium of claim 5, wherein the debug program further comprises, execution instructions for uninterruptedly executing the application until a user-specified breakpoint is encountered N-X times, wherein the execution instructions are configured for execution in response to a user request specifying the user-specified breakpoint and a number of encounters N-X, where N is a stored number of times the particular breakpoint was

encountered during the test run and X is a value equal to or greater than zero <u>and less than N</u>.

7. (Currently Amended) The <u>tangible</u> computer readable <u>storage</u> medium of claim 5, wherein the debug program further comprises, execution instructions for, after the application is stopped at a last stopped location of the test run in response to encountering a last encountered breakpoint Y number of times, processing a user-input request to uninterruptedly execute the application until the application is again stopped at the location in response to again encountering the last encountered breakpoint Y number of times.

8. (Canceled) The computer readable medium of claim 5, wherein the counting instructions, when executed:

increment a breakpoint-specific counter each time a breakpoint associated with the breakpoint-specific counter is encountered in a particular code segment; and reset the breakpoint-specific counter each time the particular code segment is entered.

9. (Previously Presented) A method of debugging an application in a debugging environment comprising the application and a debugger program, the method comprising:

associating a counter with a breakpoint and with at least one application code segment in which the breakpoint is located;

incrementing the counter each time the breakpoint is encountered, wherein the counter, at any time, always reflects a current number of times a given breakpoint has been encountered and wherein incrementing the counter is not limited by a predetermined number representative of a desired number of encounters of the given breakpoint; and

resetting the counter each time the application code segment is entered.

10. (Original) The method of claim 9, further comprising storing a counter value of the counter for use in a subsequent execution of the application.

11. (Original) The method of claim 9, wherein resetting the counter comprises firing an internal breakpoint which does not call a user interface.

12. (Original) The method of claim 9, further comprising:

determining whether a counter value of the counter has reached a user-specified value; and if so, halting execution of the application and issuing a user notification indicative of the counter value.

13. (Original) The method of claim 9, further comprising:

uninterruptedly executing the application until a counter value of the counter has reached a user-specified value; and then halting execution of the application.

14. (Currently Amended) A <u>tangible</u> computer readable <u>storage</u> medium containing a debug program which, when executed by a computer configured with an application being debugged during a debug session, performs breakpoint counter operations, the debug program comprising counter instructions which, when executed:

associate a counter with a breakpoint and with at least one application code segment in which the breakpoint is located;

increment the counter each time the breakpoint is encountered, wherein the counter, at any time, always reflects a current number of times a given breakpoint has been encountered and wherein incrementing the counter is not limited by a predetermined number representative of a desired number of encounters of the given breakpoint; and

reset the counter each time the application code segment is entered.

15. (Currently Amended) The <u>tangible</u> computer readable <u>storage</u> medium of claim 14, wherein the code segment is one of a routine and a loop.

16. (Currently Amended) The <u>tangible</u> computer readable <u>storage</u> medium of claim 14, wherein the debug program further comprises execution instructions for uninterruptedly executing the application until a counter value of the counter is equal to a user-specified value.

17. (Currently Amended) The <u>tangible</u> computer readable <u>storage</u> medium of claim 14, wherein the debug program further comprises, storing instructions for storing a counter value of the counter for use in a subsequent execution of the application.

18. (Currently Amended) The <u>tangible</u> computer readable <u>storage</u> medium of claim 14, wherein the debug program further comprises internal breakpoint setting instructions for setting an internal breakpoint configured to reset the counter without calling a user interface when the internal breakpoint is fired.

19. (Currently Amended) The <u>tangible</u> computer readable <u>storage</u> medium of claim 14, wherein the debug program further comprises counter value instructions for determining whether a counter value of the counter has reached a user-specified value and, if so, halting execution of the application and issuing a user notification indicative of the counter value.

20. (Currently Amended) The <u>tangible</u> computer readable <u>storage</u> medium of claim 14, wherein the debug program further comprises counter value instructions for allowing uninterrupted execution of the application until a counter value of the counter has reached a user-specified value and then halting execution of the application.

21. (Previously Presented) A computer system, comprising:

a memory containing content comprising at least a debug program to implement a debug session, a breakpoint table configurable with breakpoint-specific counters and an application for debugging; and a processor which, when executing at least a portion of the content during the debug session, is configured to:

associate a breakpoint-specific counter with a breakpoint and with at least one application code segment in which the breakpoint is located;

increment the counter each time the breakpoint is encountered, wherein the counter, at any time, always reflects a current number of times a given breakpoint has been encountered and wherein incrementing the counter is not limited by a predetermined number representative of a desired number of encounters of the given breakpoint; and

reset the counter each time the application code segment is entered.

22. (Original) The computer system of claim 21, wherein the debug program, when executed by the processor, configures the processor to store a counter value of the breakpoint-specific counter in the breakpoint table for use in an execution of the application.

23. (Original) The computer system of claim 21, wherein the debug program, when executed by the processor, configures the processor to set an internal breakpoint configured to reset the breakpoint-specific counter without calling a user interface when the internal breakpoint is fired.

24. (Original) The computer system of claim 21, wherein the debug program, when executed by the processor, configures the processor to uninterruptedly execute the application until a counter value of the breakpoint-specific counter is equal to a user-specified value.

25. (Original) The computer system of claim 22, wherein the debug program, when executed by the processor, configures the processor to issue a user notification indicative of the counter value.

Respectfully submitted,

Gero G. McClellan
Registration No. 44,227
PATTERSON & SHERIDAN, L.L.P.
3040 Post Oak Blvd. Suite 1500
Houston, TX 77056
Telephone: (713) 623-4844
Facsimile: (713) 623-4846
Attorney for Applicants